Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 540 – 555

doi:10.3906/elk-1812-59

# Estimation of distribution-based multiobjective design space exploration for energy and throughput-optimized MPSoCs

**Maryam MURAD[1], Ishfaq HUSSAIN[1], Ayaz AHMAD[2], Muhammad Yasir QADRI[3,*], Nadia N. QADRI[2]**

[1]Department of Electrical Engineering, HITEC University, Taxila, Pakistan
[2]Department of Electrical Engineering, COMSATS University, Islamabad, Wah Campus, Wah Cantonment, Pakistan
[3]School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK

**Received:** 09.12.2018 • **Accepted/Published Online:** 17.06.2019 • **Final Version:** 27.01.2020

**Abstract:** Modern multicore architectures comprise a large set of components and parameters that require being matched to achieve the best balance between power consumption and throughput performance for a particular application domain. The exploration of design space for finding the best power–throughput trade-off is a combinatorial optimization problem with a large number of combinations, and. in general, its solution is prohibitively difficult to be explored exhaustively. However, fortunately, evolutionary algorithms (EAs) have the potential to efficiently solve this problem with reasonable computational complexity. In this paper, we consider a multiobjective design space exploration (DSE) problem with two conflicting objectives. The first objective corresponds to power consumption minimization while the second objective relates to throughput maximization. We approach this problem by employing the estimation of distribution algorithm (EDA), which belongs to the family of EAs. The proposed EDA-based DSE (EDA-DSE) scheme efficiently selects the design parameters (i.e. cache size, number of cores, and operating frequency) with an efficient power–throughput ratio. The proposed scheme is verified using cycle-accurate simulations over a set of benchmarks and the simulation results show a significant reduction in energy-delay product for all benchmark applications when compared to the default baseline configuration and genetic algorithm.

**Key words:** Estimation of distribution algorithm, multiobjective optimization, design space exploration, energy/throughput, multiprocessal system on chip

## 1. Introduction

Multiprocessor system-on-chips (MPSoCs) have become the dominant computing paradigm in embedded and mobile processors due to their efficiency and mature programming models [1]. However, there are many challenges associated with the scalability of these architectures, among which finding the performance and energy-consumption trade-off has emerged as an important concern for applications executed on multicore architectures [2]. Advancements in this research domain promise high performance and reduced energy consumption, but with the challenge of coping with increased numbers of design options and constraints. For a general purpose computing platform, different workloads demand varying computing requirements. Therefore, the energy consumption can be minimized by adaptively varying the computing platform as per workload demand [3]. The energy dissipation in a multicore system has a significant component of cache memories as well [4]. Therefore, many researchers have taken this point into consideration in the form of "resizable" caches for different appli-

*Correspondence: yasirqadri@acm.org

cations to reduce the energy consumption with minimal effects on the performance [5]. In general, achieving the balance of performance and energy consumption on a multicore processor may be facilitated by analyzing the combination of number of cores, frequency, and cache sizes, and selecting the appropriate combination of them. It is therefore highly desirable to consider the combined impact of cache size, active number of cores, and operating frequency on the performance and energy consumption. Consequently, there is a need to devise a framework for selecting an efficient configuration of these parameters for any running application. This process of exploring multiple design options while satisfying one or more constraints is called design space exploration (DSE) [6]. The number of possible configurations in the design space may be very high and determining the best configurations for performance and energy-consumption balance by an exhaustive search may be infeasible due to high computational complexity [7]. Owing to the complexity of exhaustive searches, researchers have proposed several other techniques to utilize multicore architectures in an efficient way, such as evolutionary algorithms (EAs) [8], genetic algorithms (GAs) [9], and fuzzy logic [10].

In this paper, we employ an EA to explore the design space of multicore architecture for finding an optimal balance between two conflicting objectives, throughput maximization and energy-consumption minimization. It was demonstrated in [11] that EAs are quite suitable for the solution of multiobjective optimization problems. More specifically, the use of EAs for exploring the energy–throughput trade-off has been motivated by the fact that the population-based nature of these algorithms is capable of generating multiple configurations to determine the Pareto optimal set (trade-off solutions) in a single run. Moreover, EAs iteratively deal with a subset of possible solutions (populations) and therefore serve as the best option for multiobjective optimization problems with large search spaces [12]. In EAs, populations are initialized and evolve towards better and better solutions. For multiobjective optimization problems, EAs significantly narrow down the search space by providing a small number of potential solutions called the Pareto set [13]. Subsequently, based on the given decision criterion, designers can intelligently select the best solution from the obtained Pareto set. This process provides better exploration and selection of best trade-off among multiple objectives with reasonable computational complexity [14].

In this context, the presented work aims to devise a DSE scheme that is based on the estimation of distribution algorithm (EDA). The EDA belongs to the family of EAs, and it is based on estimating a model for the probability distribution of the various individuals/solutions in the population. After selecting the initial population (subset of possible solutions), the EDA explicitly extracts information from those selected solutions to estimate/build a probability distribution model of the promising solutions. This probability distribution model is then used for generating a new set of solutions (a population) in the next iteration. It is a relatively newer approach among evolutionary techniques and an open field for further research. The EDA has various types [15] and these types are mostly implemented for the solution of classical optimization problems such as the knapsack problem [16], traveling salesman problem [15], and nurse scheduling problem [17]. This paper, to the best of our knowledge, is the first effort to approach DSE in multicore architectures by employing the EDA.

The main contributions of this paper are:

- **Multiobjective design space exploration:** In this paper, we consider a multiobjective DSE problem for energy-consumption minimization and throughput maximization for a multicore architecture. We propose a DSE scheme that provides an optimal trade-off between these two conflicting objectives.

- **Multiple decision variables:** We consider a unified search space with the three important parameters of multicore architectures: cache size, number of cores, and frequency, and their combined impact on the selection of an efficient configuration.

- **Estimation of distribution algorithm:** We approach this multiobjective optimization problem by employing the EDA, which has the inherent advantage of probabilistic modeling.

- **Validation of results:** We have validated the final solution presented by the algorithm using a cycle-accurate simulator, Micro Architectural and System Simulator for x86 (MARSS-x86) [18].

The remaining paper is organized as follows. In Section 2, related work is presented. Section 3 provides the problem statement and overview of the EDA, and Section 4 presents the proposed EDA-based DSE scheme. In Section 5, the experimental results are illustrated, followed by the conclusion in Section 6.

## 2. Related work

The demand to improve the energy efficiency of multicore architectures without sacrificing the performance has resulted in the conception and design of various techniques. Yang et al. [19] proposed an energy-efficient scheme for mapping parallel tasks onto a multiprocessor platform. Their proposed scheme searches the ordering of tasks and the processor assignment possibilities. Based on this approach, a Pareto-optimal set is generated, in which each point is better than others in either aspect, i.e. energy efficiency or execution speed. These statistics are utilized at runtime for energy optimization by dynamic voltage scaling. Several researchers used this design for cache energy saving. Wang et al. [20] proposed a static profiling technique for exploration of different possible configurations of the two-level cache in the case of a multicore system with private L1 caches and a shared L2 cache. This technique dynamically reconfigures the L1 cache and partitions the L2 cache for energy consumption minimization. In [21], Rawlins et al. considered cache resizing for energy optimization in heterogeneous dual-core systems by altering the size of the L1 cache. The authors also addressed core interactions, data coherence, and various operational issues. All the aforementioned works devised techniques to reduce energy by cache configurations. However, there are a variety of other factors upon which energy consumption depends. Motivated by this fact, the current work provides a comprehensive approach for DSE that considers the mutual impact of different factors along with multiple objectives.

Moreover, in multicore systems where dozens of cores are involved, it is difficult to find configurations that maximize throughput with a power constraint [22]. In [23] Reagen et al. explored the design space for low-power accelerators used in SoC designs to obtain power and performance trade-off. Qadri et al. [10] proposed a scheme called E-FLORE to optimize multicore architectures. The E-FLORE technique finds a trade-off between energy and throughput using fuzzy logic. However, fuzzy-based techniques lose the computational efficiency for larger systems due to the increased number of fuzzy rules and membership functions. In order to overcome the limitations associated with fuzzy systems, hybrid systems merging fuzzy logic with EAs are also reported in the literature [24, 25]. These hybrid systems result in improved performance by optimizing the fuzzy rules and membership functions.

Optimization theory-based search methods provide another way of achieving energy efficiency and performance improvement. Among these methods, multiobjective evolutionary techniques have the potential to find a set of possible configurations while guaranteeing a certain level of closeness of this set to the actual Pareto front with significantly reduced exploration time [26]. Many multiobjective evolutionary algorithms (e.g., [11, 27–29])

have been widely reported in the literature for various applications to find the trade-off solutions to achieve the best possible compromise among multiple objectives.

Among evolutionary algorithms, the genetic algorithm [30] and ant-colony optimization [31] have been employed for DSE for multicore architecture. Palesi et al. [30] considered configuration space exploration for a SoC and extended the Platune tuning approach [32] using a genetic algorithm and exhaustive search to increase the exploration speed with many configurable parameters (e.g., voltage, capacitance). Wang et al. [31] used ant-colony optimization for design space exploration to minimize the completion time of the applications while effectively utilizing the computational resources. This method finds a trade-off between hardware cost and timing performance. Ferrandi et al. [33] proposed an ant-colony based DSE to optimize the performance of an MPSoC. This approach explores different designs to decide task mapping and establish the routine to execute tasks. It determines efficient hardware/software partitioning using hierarchical task graphs for reconfigurable heterogeneous MPSoC applications.

The main objective of this paper is to make a trade-off between energy consumption and throughput by selecting an appropriate combination of cores, cache size, and operating frequency. There exist various mathematical models for energy and throughput evaluation for computing platforms [34–36]. In the proposed design of the EDA-based DSE scheme, we use the model presented by Qadri et al. [37]. Employing the EDA, our scheme evolves by estimating the probability distribution over the search space depending on how the individuals are distributed in the population [38], which makes it different from other classical DSE techniques. The transparency of the probabilistic modeling of the EDA provides better guidelines for the solution search process. It has a better mathematical foundation as compared to other evolutionary algorithms. It has a faster convergence rate and makes better use of fitness information of previous generation [39].

The principle advantage of the EDA over genetic algorithms is its property of being free of multiple recombination methods such as mutation or crossover. The recombination and evolution methods of genetic algorithms sometimes result in premature convergence and many important solutions are disrupted. These issues can be addressed in the EDA, where new solutions are generated according to the probability estimation of the previous generation [17]. Compared to ant-colony optimization, which is considered to be suitable for discrete optimization problems, the EDA gives a vast variety of models for both discrete [40] and continuous cases [41]. In particle swarm optimization, parameters are always interdependent, while the EDA is a probabilistic model for both independent variables [42] (i.e. in the form of univariate marginal distribution [43]) and dependent variables.

## 3. DSE problem and overview of EDA

### 3.1. DSE problem statement

For modern computer architects, energy efficiency has been the top design objective along with the ever increasing demand of throughput [44]. Keeping that in view, the two basic objectives of this paper are to efficiently select a system configuration in order to minimize the energy consumption and to maximize the throughput of the system. Performance and energy balance on multicore processors may be facilitated by analyzing combinations of numbers of cores, frequency, and cache sizes. The impact of these three variables is summarized as follows:

**Cache size:**

Processor performance has surpassed the memory throughput for more than a decade. The memory wall is usually bridged by introducing a hierarchy of memories, i.e. small and fast cache levels. Smaller cache sizes

may result in increased capacity misses and large cache size leads to lower response time. Therefore, selecting an appropriate cache size is critical for energy/throughput balance.

**Frequency:**

Processor power dissipation directly depends on the operating frequency. Dynamic voltage frequency scaling (DVFS) is a method used for power reduction that involves operating the processor at different frequencies as per the need of performance and energy constraints. Selecting a suitable frequency is imperative for energy- and throughput-constrained systems.

**Number of cores:**

Performance of multicore systems does not scale with the number of cores as defined by Amdhal's law [45]. However, adding cores does contribute to the energy consumption of an architecture. Therefore, having a balanced number of cores is a key factor to be considered along with other parameters to design a high-performance energy-efficient system.

Therefore, the decision/design variables considered in this paper include the aforementioned parameters, i.e. number of cores, cache size, and operating frequency.

In order to define the interdependence of the aforementioned objectives, an objective function is required to be specified. Qadri et al. [37] presented energy and throughput models that not only incorporated the said design variables but also defined a simplistic and linear relationship of them with the energy and throughput of a multicore system. The models are found to be accurate for up to 5% when compared with highly accurate simulation models over a number of benchmark applications.

Energy consumption minimization and throughput maximization are two conflicting objectives. That is, finding the optimal (best possible) solution for energy minimization may result in a nonoptimal solution to throughput maximization and vice versa. Therefore, a solution that achieves a balance between these conflicting objectives is needed. To this end, we devise an EDA-based exploration framework that guides us towards a set of efficient configurations (i.e. Pareto optimal solutions). The solution of a biobjective optimization is called Pareto optimal if no other solution exists that could make one of the objectives better without making the other one worse. In the context of this paper, a solution that comprises the values of cache size, operating frequency, and number of cores is Pareto optimal if no other solution exists that improves energy efficiency without decreasing the throughput or increasing the throughput without decreasing the energy efficiency. We evaluate the performance of our framework using the MARSS-x86 simulator [18] by applying the obtained configurations on benchmark applications.

### 3.2. Overview of EDA

The EDA is an emerging and promising area of research in evolutionary computation, which was introduced for the first time in [43]. Researchers have proposed a number of ways in which the EDA can be implemented based on different probabilistic models. Some variants are discrete while others are either continuous or a combination of the two. Among these proposed designs, the most common are the UMDA [46], Bayesian optimization algorithm [47], and factorization distributional algorithm (FDA)[48]. Being suitable for our discrete DSE problem and based on a simple underlying probabilistic model with easy implementation, we use the UMDA in the development of our DSE framework. Throughout this paper, the term "EDA" is used as a synonym for UMDA.

Algorithm 1 presents the general pseudocode for the EDA. The algorithm starts by initializing the population.

**Algorithm 1** Pseudocode for EDA.

```
Initialize the population
while termination criterion is not met do
    Evaluate the population using fitness function
    Select a set of best individuals based on a specific selection method
    Calculate a probabilistic model from the selected set of individuals
    Generate new population by using the probabilistic model
end while
```

For the initial population, fitness values are calculated. A set of individuals with higher fitness values are selected from the initial population. Then, from this set of individuals, a probabilistic model is determined and the new population is generated. This newly generated population is considered in the next iteration. The algorithm stops when a certain termination criterion is met. Generally, the termination criterion is the maximum number of generations. However, there can be other termination criteria. For example, when there is no significant change in the fitness function values of two consecutive iterations, then the algorithm is stopped [15].

The basic concept of the UMDA, which was modeled by Muhlenbein [43], has been used in our paper. In this particular type of EDA, first, as it uses the integers 1 and 0 to represent bits in a binary string representation, the individuals values (solutions) are converted into binary form, and then, for each bit position, the frequency of occurrence of binary ones for all the individuals is computed. This determines the probability distribution vector, which is used for the generation of new individuals.

## 4. The proposed ED-based DSE scheme

The proposed design search space exploration scheme uses the EDA to find a set of combinations of cache sizes, numbers of cores, and frequencies to get efficient energy and throughput values. The selection of the design variables has been discussed in Section 3. The EDA, being part of the family of EAs, is different from traditional ones in the way it produces offspring from parents. In contrast to traditional EAs, the EDA estimates a probability distribution over the search space based on how the parent individuals are distributed in the search space, and then it samples the offspring individuals from this distribution [15]. It has been suggested that in certain problem domains the estimated probability distribution can make the search more effective and reduce the overall runtime [49], [38].

The pseudocode of the proposed EDA-DSE technique is summarized in Algorithm 2. The algorithm takes the population size, number of generations, and solution space as inputs and returns the set of nondominated solutions (Pareto front) as the output. The major steps of the algorithm are described as follows:

- **Initialization, evaluation, and sorting (lines 1–4):** In these steps, first, the initial population, $S_0$ of size $L$, is randomly generated from the given solution space. Then each individual $s_i \in S_0$ is evaluated, and their energy consumption and throughput are computed. The evaluated set of individuals is then arranged in a descending pattern with the best solution at the top using two attributes, which are inspired by the fast sort algorithm[50]. Then a rank value and a crowding distance value are assigned to it. The rank values, $R_0$, and crowding distance values, $C_0$, are computed by using nondominated sorting and ranking and crowding distance methods [28], respectively.
  - *Nondominated sorting and ranking:* Population $S_0$ is sorted based on a nondomination technique into different Pareto fronts. The first Pareto front represents the completely nondominated set

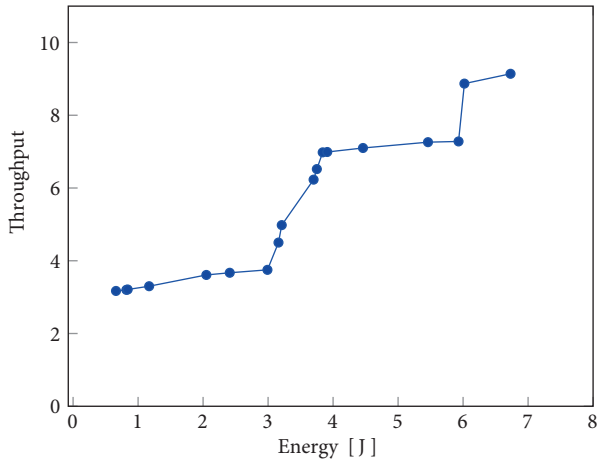


**Figure 1**. ED-DSE Algorithm

**Algorithm 2** Estimation of distribution algorithm.

**Input:** Population size ($L$), number of generations ($G$), solution space
$S_0 \leftarrow$ Generate initial population randomly from the given solution space
$R_0, C_0 \leftarrow$ Evaluate each individual $s_i \in S_0$ and compute its rank and crowding distance
$N_0 \leftarrow$ Based on $R_0$ and $C_0$, arrange the individuals of $S_0$ in descending pattern
$j \leftarrow 0$
**while** termination criterion is not met **do**
    $N_j^{sub} \leftarrow$ Select a subset of best individuals from $N_j$
    $P_j \leftarrow$ Estimate the probability of each solution in $N_j^{sub}$
    $O_j \leftarrow$ Generate offspring by sampling new individuals according to $P_j$
    $S_j \leftarrow N_j^{sub} \cup O_j$, Update the population
    $R_{j+1}, C_{j+1} \leftarrow$ Update $R_j$ and $C_j$ according to $S_j$
    $N_{j+1} \leftarrow$ Arrange the individuals of $S_j$ in descending pattern according to $R_j$ and $C_j$
    $j \leftarrow j + 1$
**end while**
**Output:** Nondominated solutions set (first Pareto front)

of individuals in the current population. The second Pareto front contains individuals that are dominated by the individuals in the first front only, whereas the individuals in the third Pareto front are dominated by the individuals of the first and second Pareto fronts, and so on. In order to identify solutions of the first nondominated front in a population, each solution can be compared with every other solution in the population to find out if it is dominated or not. At this stage, all individuals in the first nondominated front are found. Each individual in the front are assigned a fitness value, which is marked as the rank. The rank of an individual represents the front number to which it belongs. Between two individuals with differing nondomination ranks, the individual with the lower rank is preferred. If both individuals belong to the same front, then the crowding distance is used for comparison.

– *Crowding distance calculation:* Once the nondominated sorting is completed, a crowding distance is assigned to each individual. The total crowding distance of an individual is the sum of its objectives' distances, which are the absolute normalized differences between the values of the individual and its closest neighbors. Since the individuals are selected based on the rank and crowding distance, all

the individuals in the population are assigned a crowding distance value. The crowding distances are assigned based on Pareto front.

Finally, the evaluated set of individuals is arranged in a descending pattern in $N_0$ with the best individual at the top according to the two attributes, i.e. rank and crowding distance. An individual is preferred over others if its rank is less than the others or if its crowding distance is greater than the others. Thus, the selection is based both on ranks and crowding distances. The generation/iteration number, $j$, is initialized to 0.

- **Selection (line 6):** A subset of best individuals $N_j^{sub}$ is selected from the ordered set $N_j$. The algorithm uses a truncation selection criterion where the top 50% best individuals are selected and the rest are truncated.

- **Probability distribution estimation (line 7):** In this step, the probability distribution of the individuals in $N_j^{sub}$ is estimated. The variables of each individual (comprising of cache size ($ca$), number of cores ($co$) and operating frequency ($fr$)) in the selected set $N_j^{sub}$ are converted into binary integers and stored in an $n+m+k$ dimensional vector $\mathbf{x} = [x_1 x_2 ... x_{k+m+n}]$ as follows:

$$\begin{aligned} \mathbf{x} =: [ca, co, fr] = [b_1^{ca} b_2^{ca} ... b_k^{ca}, b_1^{co} b_2^{co} ... \\ b_m^{co}, b_1^{fr} b_2^{fr} ... b_n^{fr}]. \end{aligned} \tag{1}$$

The first $k$ bits of $\mathbf{x}$ represent the cache size, the $(k+1)$th to $(k+m)$th bits denote the number of cores, and the bits from the $(k+m+1)$th to the $(k+m+n)$th position represent the value of operating frequency. The probability distribution of $x_i$ is estimated as follows:

$$p_i = \sum_{l=1}^{L} \frac{I_{l,i}}{L}, \tag{2}$$

where $I_{l,i}$ is the indicator function that is equal to the value of $x_i$ in the $l$th individual, defined by:

$$I_{l,i} = \begin{cases} 1 & x_i = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The estimated probability density values, $p_i$, are stored in $P_j$.

- **Offspring generation (line 8):** Offspring, $O_j$, are generated by sampling new individuals according to $P_j$. The following sampling technique that is inspired by [51] is used for offspring generation:

$$x_i = \begin{cases} 1 & \text{if } random(0,1) \leq p_i \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The variables in each offspring are converted from binary to integer values. The number of individuals in $O_j$ is equal to $L/2$ to keep the size of the new population as initial $L$.

- **Replacement, evaluation, and sorting (lines 9–12):** The generated offspring, $O_j$, and the selected parents, $N_0^{sub}$, are then merged together in $S_j$; their ranks and crowding distances ($R_j$ and $C_j$) are recomputed; and the individuals in $S_j$ are arranged in a descending pattern. The resulting arranged set $N_{j+1}$ forms the new population for another iteration of the algorithm.
- **Termination (line 14):** In our case, the stopping criterion is the maximum number of generations, $G$. When this condition is met (i.e. $j = G$), the first Pareto front is saved and the algorithm ends.

## 5. Simulation results and discussion

In this section, we present the simulation results and demonstrate the performance of our proposed EDA-DSE algorithm.

### 5.1. Simulation setup for EDA-DSE

We first execute the proposed EDA-DSE algorithm in MATLAB, which gives us the Pareto front (solution set) of our multiobjective DSE problem. From the obtained Pareto front, we select the solution with maximum throughput and the solution with minimum energy consumption, and we analyze their results using the MARSS-x86 simulator to validate the performance of the proposed EDA-DSE. In the rest of the paper, these solutions will be called the throughput-efficient (TE) and energy-efficient (EE) solutions, respectively.

Marss-x86 is a cycle-accurate simulator and is used for multicore implementations. It is built on QEMU to support cycle-accurate simulations of multicore x86 processors. MARSS-x86 includes detailed and cycle-accurate models of coherent caches for single and multicore processor chips, interconnections, chipsets, DRAM memory, and IO devices.

We used the Stanford Parallel Applications for Shared Memory (SPLASH-2) benchmarks for the evaluation of the proposed configuration. The TE and EE solutions are evaluated in MARSSx86 for five applications of SPLASH-2, i.e. Barnes, FMM, Ocean, Water-Spatial, and LU. HP Lab's CACTI tool [52] is used to get the per access energy for various cache sizes of both L1 and L2.

#### 5.1.1. Benchmarks details

Following is a brief overview of the SPLASH-2 benchmarks [53] used to evaluate the proposed configurations:

- Barnes (BRN): The Barnes application is simulated by the Barnes–Hut hierarchical N-body method. It has a problem size of 16K particles.
- FMM (FMM): This benchmark simulates by fast multiple method [54] with interaction in two dimensions. The problem size for FMM is 16K particles.
- Ocean (OCN): The Ocean application studies large-scale ocean-type movements and has a problem size of 258 $\times$ 258.
- Water-Spatial (WTR): This application evaluates potentials over time in a system O(n) algorithm and uses a uniform 3D grid of cells. Its problem size is 512 molecules.
- LU (LU): The LU kernel factorizes a dense matrix into the product of a lower triangular and an upper triangular matrix and has a problem size of a 512 $\times$ 512 matrix with block size of 16.

### 5.2. Algorithm results

We simulate the EDA-DSE algorithm for a generation size of up to 30 and a population size of 100. The simulation provides the Pareto front (solution set) of the multiobjective DSE problem. The results were analyzed based on different generation sizes of 5, 10, 15, 20, and 25. The results showed that, in general, with increasing generation and population sizes, energy is reduced and throughput is increased. However, by increasing the generation size above 30 and the population size above 100, there is no notable increase in energy efficiency and throughput performance. Furthermore, by increasing the size of generations and/or the population, more time will be taken by the algorithm. Therefore, we have kept the generation and population sizes as 30 and 100, respectively.

Table 1 shows the Pareto-efficient values of the three decision variables of cache size, number of cores, and frequency along with energy consumption and throughput associated with each of the Pareto-efficient solutions and Figure 1 shows the obtained Pareto front. Thus, an efficient and acceptable solution can be selected from the obtained Pareto front by using an expert opinion.

**Table 1**. Pareto-optimal results for generation size of 30.

| $No.of cores$ | $Cache sizes[KB]$ | $Frequency[Hz]$ | $Energy[J]$ | $Throughput*$ |
|---|---|---|---|---|
| 15 | 16 | 53 | 0.2380 | 0.3917 |
| 2 | 64 | 75 | 0.0444 | 0.2291 |
| 13 | 64 | 62 | 0.3721 | 0.4379 |
| 10 | 8 | 40 | 0.0474 | 0.2695 |
| 2 | 256 | 79 | 0.1271 | 0.3895 |
| 9 | 64 | 62 | 0.2572 | 0.4184 |
| 4 | 64 | 67 | 0.1109 | 0.3461 |
| 6 | 128 | 70 | 0.4760 | 0.5102 |
| 2 | 16 | 71 | 0.0131 | 0.1742 |
| 12 | 256 | 80 | 1.0932 | 0.7494 |

*Throughput here is a unitless quantity, since it is normalized to the throughput of a full-scale configuration, i.e. 16 cores, 256 KB cache, and 80 MHz clock frequency.

### 5.3. Solution selection and results validation

To verify the performance of the proposed EDA-DSE scheme over MARSS-x86, we select two solutions: the solution with the lowest energy consumption (EE solution) and the solution with the highest throughput (TE solution) from Table 1. Referring to the table, the solution with the combination of 2 cores, 16 KB cache size, and 71 MHz frequency consumes 0.013 J of energy and therefore it is the EE solution. The solution with 12 cores, 256 KB cache size, and 80 MHz frequency is the TE solution as it obtains about 75% of the throughput when normalized against the default configuration, which is the highest throughput by any solution).

In order to check the efficiency of the solution, the results of the EDA-DSE exploration engine are compared with the default configuration and GA (see Table 2) by using the methodology discussed in [55]. For the GA, design variables and the objective function are kept the same as for the EDA. Generation size of 30 and population size of 100 were taken for the GA for comparison.

For each of the two algorithms, two solutions (i.e. EE and TE) are presented. The obtained results are compared

in Figure 2 for the suggested solution. See Figure 2a for normalized throughput, Figure 2b for normalized energy consumption, and Figure 2c for energy delay product.

The combinations associated with the aforementioned two solutions are analyzed with various SPLASH-2 benchmark applications, i.e. Barnes, FMM, Ocean, Water-Spatial, and LU. Figure 2 presents the results obtained for different parameters as follows:

1. Normalized throughput: The benchmarks results for 12 cores, which are TE, show a minor throughput decrease of 2% to 4% for Barnes, Ocean, and Water-Spatial, whereas for the rest of the applications the throughput remains the same as the default configuration (see Figure 2a). This is due to the fact that according to Amdahl's law [45] for a parallel application the throughput saturates after a limited number of cores. For example, if we assume an application to be 95% parallel, then as per Amdahl's law for 16 cores the speed-up will be 1.79 and for 12 cores it will be 1.73, i.e. only a 2.2% increase in throughput. Therefore, increasing the number of cores beyond a certain limit may not result in significant throughput increase for a given amount of parallelism in an application.

   It is quite clear in Figure 2a that the estimation of distribution for the TE solution (i.e. ED − TE) is better than that of the GA (i.e. GE − TE). For the LU benchmark the ED − TE solution is 42% more efficient than GE − TE. In the case of FMM, Ocean, and Water Spatial, this difference is 5%, 16%, and 3%, respectively, while for Barnes the ED − TE solution throughput is the same as that of the GE − TE solution.

   Furthermore, Figure 2a shows that the EE configuration results in the lowest throughput drop of no more than 55% in the case of Barnes, and for the rest of the applications the throughput remains not less than 50% of that of the default configuration. Along with remarkable reduction in energy consumption, throughput values are also better than the GA for some benchmark applications (i.e. Water-Spatial and LU), while they are slightly lower for Barnes, FMM, and Ocean. Even if the average throughput for both algorithms is considered, the suggested ED − DSE solution is 10.4% better than that of the GA.

2. Normalized energy: In order to check the efficiency of our suggested algorithm, results were compared with the default configuration and GA. Figure 2b shows normalized energy of the default configuration, EDA, and GA. While comparing the aggregate core energies of 12 cores (ED − TE) and 2 cores (ED − EE) a reduction in energy consumption by 22% to 25% for all applications in the case of 12 cores, the TE configuration, can be observed. However, it can be seen that if we compare the TE solution suggested by the EDA with the GA counterpart, energy consumption values are slightly higher. Energy consumption values vary between 0.75 to 0.78 while for the GA these are in the range of 0.46 to 0.77. Overall, there is an average difference of 30%. However, for the EE solution, the EDA outperforms both the GA and default configuration. All the applications show a significant reduction in energy consumption, i.e. between 77% and 82% for the default configuration and 42% to 88% for the GA. As can be seen in Figure 2b, energy consumption values for all the benchmarks are lower compared to the GA. On average the suggested ED − DSE solution uses 85.5% less energy as compared to the GA solution.

3. Normalized EDP: Figure 2c shows the ratio of normalized energy delay product (EDP) between the default configuration, EDA solution (i.e. ED − TE, ED − EE) and GA (i.e. GE − TE, GE − EE). EDP is a commonly used metric to assess the trade-off between the energy and throughput of an architecture [56]. The lower the EDP, the more the configuration will be efficient in terms of both energy and throughput.

**Table 2**. The three configurations: default, throughput-efficient, and energy-efficient (EE).

| Configuration | Number of cores | Cache size [KB] | Frequency [MHz] |
|---|---|---|---|
| Default | 16 | 256 | 80 |
| Throughput-efficient | 12 | 256 | 80 |
| Energy-efficient | 2 | 16 | 71 |

For the EDA TE (ED − TE) configuration, the EDP remains between 75% and 82% that of the default configuration. On the other hand, it is slightly greater than that of GE − TE. This difference is between 20% to 31%. For the EDA EE (ED − EE) configuration, the lowest amount of EDP could be observed for LU and Water-Spatial, i.e. 16% and 20%, respectively, while compared with GE − EE, ED − EE has remarkably less normalized energy delay product, i.e. the difference is between 33% and 84%.

Therefore, it can be concluded that the proposed estimation distribution-based DSE scheme can successfully propose EE and TE solutions for a multicore architecture. Hence, it could be used at design time to recommend optimal and efficient configurations in terms of throughput and energy consumption.

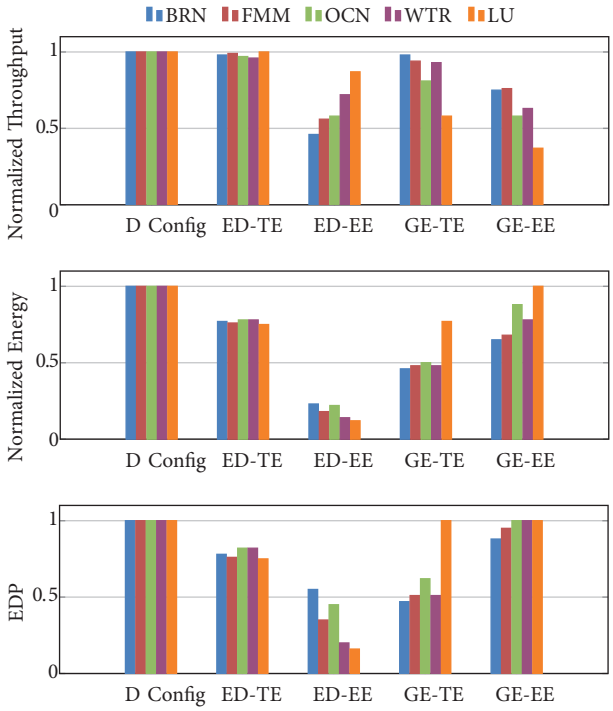


**Figure 2**. Benchmarks results: (a) normalized throughput, (b) normalized energy, and (c) EDP.

## 6. Conclusion

In this paper, we have proposed a multiobjective DSE framework for multicore architectures using the EDA. The proposed framework searches the set of best configurations with three decision variables, number of cores, L1 cache size, and CPU operating frequency, in order to minimize the energy consumption and maximize the throughput. The results of the proposed framework were analyzed using SPLASH-2 benchmark applications integrated into the MARSS-x86 simulator. The simulation results show that the EDA-based DSE framework can

efficiently address the DSE problem in multicore architectures by achieving a better trade-off between energy consumption and throughput.

**Acknowledgment**

This work was supported by the National ICT R&D Fund Pakistan through grant number ICTRDF/TR&D /2012/65.

## References

[1] Palermo G, Silvano C, Zaccaria V. ReSPIR: A response surface-based Pareto iterative refinement for application-specific design space exploration. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2009; 28 (12): 1816-1829.

[2] Balasubramonian R, Albonesi D, Buyuktosunoglu A, Dwarkadas S. Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures. In: Proceedings of the 33rd Annual ACM/IEEE International Symposium on Microarchitecture; New York, NY, USA; 2000. pp. 245-257.

[3] Korthikanti VA, Agha G. Energy-performance trade-off analysis of parallel algorithms. In: USENIX Workshop on Hot Topics in Parallelism; Berkeley, CA; 2010. pp. 106-116.

[4] Mittal S, Cao Y, Zhang Z. Master: A multicore cache energy-saving technique using dynamic cache reconfiguration. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2014; 22 (8): 1653-1665.

[5] Yang SH, Powell MD, Falsafi B, Vijaykumar T. Exploiting choice in resizable cache design to optimize deep-submicron processor energy-delay. In: Proceedings of the Eighth International Symposium on High-Performance Computer Architecture; Boston, MA, USA; 2002. pp. 151-161.

[6] Monchiero M, Canal R, Gonzalez A. Design space exploration for multicore architectures: a power/performance/thermal view. In: Proceedings of the 20th Annual International Conference on Supercomputing; Queensland, Australia; 2006. pp. 177-186.

[7] Calborean H, Vinan L. Multi-objective optimization of advanced computer architectures using domain-knowledge. PhD, Lucian Blaga University of Sibiu, Sibiu, Romania, 2011.

[8] Calborean H, Jahr R, Ungerer T, Vintan L. Optimizing a superscalar system using multi-objective design space exploration. In: Proceedings of the 18th International Conference on Control Systems and Computer Science; Bucharest, Romania; 2011. pp. 339-346.

[9] Mariani G, Avasare P, Vanmeerbeeck G, Ykman-Couvreur C, Palermo G et al. An industrial design space exploration framework for supporting run-time resource management on multi-core systems. In: Design, Automation & Test in Europe Conference & Exhibition (DATE); Dresden, Germany; 2010. pp. 196-201.

[10] Qadri MY, McDonald Maier KD, Qadri NN. Energy and throughput aware fuzzy logic based reconfiguration for MPSoCs. Journal of Intelligent and Fuzzy Systems 2014; 26 (1): 101-113.

[11] Deb K. Multi-objective Optimization Using Evolutionary Algorithms. Vol. 16. Chichester, UK: John Wiley & Sons, 2001.

[12] Coello CAC, Van Veldhuizen DA, Lamont GB. Evolutionary Algorithms for Solving Multi-objective Problems. Vol. 242. New York, NY, USA: Springer, 2002.

[13] Yang XS. Metaheuristic optimization: algorithm analysis and open problems. arXiv preprint. arXiv: 12120220. 2012.

[14] Fonseca CM, Fleming PJ. An overview of evolutionary algorithms in multiobjective optimization. Evolutionary Computation 1995; 3 (1): 1-16.

[15] Larrañaga P, Lozano JA (editors). Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Vol. 2. Boston, MA, USA: Springer, 2002, pp. 129-145.

[16] Li H, Zhang Q, Tsang E, Ford JA. Hybrid estimation of distribution algorithm for multiobjective knapsack problem. In: Evolutionary Computation in Combinatorial Optimization: 4th European Conference; Coimbra, Portugal; 2004. pp. 145-154.

[17] Aickelin U, Li J. An estimation of distribution algorithm for nurse scheduling. Annals of Operations Research 2007; 155 (1): 289-309.

[18] Patel A, Afram F, Ghose K. Marss-x86: A Qemu-based micro-architectural and systems simulator for x86 multicore processors. In: First International Qemu Users Forum; Grenoble, France; 2011. pp. 29-30.

[19] Yang P, Marchal P, Wong C, Himpe S, Catthoor F et al. Managing dynamic concurrent tasks in embedded real-time multimedia systems. In: Proceedings of the 15th International Symposium on System Synthesis; Kyoto, Japan; 2002. pp. 112-119.

[20] Wang W, Mishra P. Dynamic reconfiguration of two-level caches in soft real-time embedded systems. In: IEEE Computer Society Annual Symposium on VLSI; Florida, USA; 2009. pp. 145-150.

[21] Rawlins M, Gordon-Ross A. CPACT-The conditional parameter adjustment cache tuner for dual-core architectures. In: 2011 IEEE 29th International Conference on Computer Design; Amherst, MA, USA; 2011. pp. 396-403.

[22] Petrica P, Izraelevitz AM, Albonesi DH, Shoemaker CA. Flicker: a dynamically adaptive architecture for power limited multicore systems. ACM SIGARCH Computer Architecture News 2013; 41; 13-23.

[23] Reagen B, Shao YS, Wei GY, Brooks D. Quantifying acceleration: Power/performance trade-offs of application kernels in hardware. In: Proceedings of the International Symposium on Low Power Electronics and Design; Beijing, China; 2013. pp. 395-400.

[24] Dennis B, Muthukrishnan S. AGFS: Adaptive genetic fuzzy system for medical data classification. Applied Soft Computing 2014; 25: 242-252.

[25] Kerre EE, Nachtegael M. Fuzzy Techniques In Image Processing. Vol. 52. Heidelberg, Germany: Springer Science & Business Media, 2000.

[26] Silvano C, Fornaciari W, Palermo G, Zaccaria V, Castro F et al. MULTICUBE: Multi-objective design space exploration of multi-core architectures. In: VLSI 2010 Annual Symposium; Lixouri, Greece; 2010. pp. 47-63.

[27] Borges CC, Barbosa HJ. A non-generational genetic algorithm for multiobjective optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation; California, USA; 2000. pp. 172-179.

[28] Srinivas N, Deb K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 1994; 2 (3): 221-248.

[29] Ghezail F, Pierreval H, Hajri-Gabouj S. Multi objective optimization using ant colonies. In: Bertelle C, Duchamp CGHE, Kadri-Dahmani H (editors). Complex Systems and Self-organization Modelling. Berlin, Germany: Springer, 2009, pp. 65-70.

[30] Palesi M, Givargis T. Multi-objective design space exploration using genetic algorithms. In: Proceedings of the Tenth International Symposium on Hardware/Software Codesign, 2002; Colorado, USA; 2002. pp. 67-72.

[31] Wang G, Gong W, DeRenzi B, Kastner R. Design space exploration using time and resource duality with the ant colony optimization. In: Proceedings of the 43rd Annual Design Automation Conference; New York, NY, USA; 2006. pp. 451-454.

[32] Givargis T, Vahid F. Platune: A tuning framework for system-on-a-chip platforms. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2002; 21 (11): 1317-1327.

[33] Ferrandi F, Lanzi PL, Pilato C, Sciuto D, Tumeo A. Ant colony optimization for mapping, scheduling and placing in reconfigurable systems. In: 2013 NASA/ESA Conference on Adaptive Hardware and Systems; Torino, Italy; 2013. pp. 47-54.

[34] Li S, Ahn JH, Strong RD, Brockman JB, Tullsen DM et al. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In: Proceedings of International Symposium on Microarchitecture; New York, NY, USA; 2009. pp. 469-480.

[35] Kamble MB, Ghose K. Analytical energy dissipation models for low power caches. In: Proceedings International Symposium on Low Power Electronics and Design; Monterey, CA, USA; 1997. pp. 143-148.

[36] Qadri MY, McDonald-Maier KD. Analytical evaluation of energy and throughput for multilevel caches. In: 12th International Conference on Computer Modelling and Simulation; Cambridge, UK; 2010. pp. 598-603.

[37] Qadri MY, McDonald-Maier KD. Data cache-energy and throughput models: design exploration for embedded processors. EURASIP Journal on Embedded Systems 2009; 2009: 725438.

[38] Chen T, Lehre PK, Tang K, Yao X. When is an estimation of distribution algorithm better than an evolutionary algorithm? In: 2009 IEEE Congress on Evolutionary Computation; Trondheim, Norway; 2009. pp. 1470-1477.

[39] Dikmen O, Akın HL, Alpaydın E. Estimating distributions in genetic algorithms. In: International Symposium on Computer and Information Sciences; Antalya, Turkey; 2003. pp. 521-528.

[40] Očenášek J, Schwarz J. The parallel Bayesian optimization algorithm. In: The State of the Art in Computational Intelligence, Proceedings of the European Symposium on Computational Intelligence; Košice, Slovakia; 2000. pp. 61-67.

[41] Costa M, Minisci E. MOPED: a multi-objective Parzen-based estimation of distribution algorithm for continuous problems. In: International Conference on Evolutionary Multi-Criterion Optimization; Faro, Portugal; 2003. pp. 282-294.

[42] Barros M, Barros MF, Guilherme J, Horta N. Analog Circuits and Systems Optimization Based on Evolutionary Computation Techniques. Erfurt, Germany: Springer, 2010.

[43] Mühlenbein H, Bendisch J, Voigt HM. From recombination of genes to the estimation of distributions II. Continuous parameters. In: Parallel Problem Solving from Nature; Berlin, Germany; 1996. pp. 188-197.

[44] Blake G, Dreslinski RG, Mudge T. A survey of multicore processors. IEEE Signal Processing Magazine 2009; 26 (6): 26-37.

[45] Hill MD, Marty MR. Amdahl's law in the multicore era. Computer 2008; 41 (7): 33-38.

[46] Wu Y, Wang Y, Liu X. Multi-population based univariate marginal distribution algorithm for dynamic optimization problems. Journal of Intelligent & Robotic Systems 2010; 59 (2): 127-144.

[47] Pelikan M. Hierarchical Bayesian optimization Algorithm. Berlin, Heidelberg: Springer, 2005.

[48] Mühlenbein H, Mahnig T. FDA-A scalable evolutionary algorithm for the optimization of additively decomposed functions. Evolutionary Computation 1999; 7 (4): 353-376.

[49] Pelikan M, Sastry K, Goldberg DE. Scalability of the Bayesian optimization algorithm. International Journal of Approximate Reasoning 2002; 31 (3): 221-258.

[50] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2002; 6 (2): 182-197.

[51] Shim VA, Tan KC, Cheong CY. A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 2012; 42 (5): 682-691.

[52] Muralimanohar N, Balasubramonian R, Jouppi NP. CACTI 6.0: A tool to model large caches. HP Laboratories 2009; 1: 1-24

[53] Woo SC, Ohara M, Torrie E, Singh JP, Gupta A. The SPLASH-2 programs: characterization and methodological considerations. SIGARCH Computer Architecture News 1995; 23: 24-36.

[54] Bailey DH. FFTs in external of hierarchical memory. In: Proceedings of the 1989 ACM/IEEE Conference on Supercomputing; New York, NY, USA; 1989. pp. 234-242.

[55] Subtil RF, Carrano EG, Souza MJ, Takahashi RH. Using an enhanced integer NSGA-II for solving the multiobjective generalized assignment problem. In: IEEE Congress on Evolutionary Computation; Barcelona, Spain; 2010. pp. 1-7.

[56] Eeckhout L, Nussbaum S, Smith JE, De Bosschere K. Statistical simulation: adding efficiency to the computer designer's toolbox. IEEE Micro 2003; 23 (5): 26-38.

[57] Fernandes CM, Merelo J, Ramos V, Rosa AC. A self-organized criticality mutation operator for dynamic optimization problems. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation; New York, NY, USA; 2008. pp. 937-944.